Workload Characterization for MG-RAST Metagenomic Data Analytics Service in the Cloud

Wei Tang,* Jared Bischof,†* Narayan Desai,‡* Kanak Mahadik,§ Wolfgang Gerlach,†*
Travis Harrison,†* Andreas Wilke,*† Folker Meyer*†

*Argonne National Laboratory, Argonne, IL, USA
†University of Chicago, Chicago, IL, USA
‡Ericsson, San Jose, CA, USA
§Purdue University, West Lafayette, IN, USA
{wtang, jbischof, wgerlach, teharrison, wilke, folker}@mcs.anl.gov*†
narayan.desai@ericsson.com,‡ kmahadik@purdue.edu§

***Abstract*—The cost of DNA sequencing has plummeted in recent years. The consequent data deluge has imposed big burdens for data analysis applications. For example, MG-RAST, a production open-public metagenome annotation service, has experienced increasingly large amount of data submission and has demanded scalable resources for the computational needs. To address this problem, we have developed a scalable platform to port MG-RAST workloads into the cloud, where elastic computing resources can be used on demand. To efficiently utilize such resources, however, one must understand the characteristics of the application workloads. In this paper, we characterize the MG-RAST workloads running in the cloud, from the perspectives of computation, I/O, and data transfer. Insights from this work will help guide application enhancement, service operation, and resource management for MG-RAST and similar big data applications demanding elastic computing resources.**

***Keywords*-Big data applications, bioinformatics, workload characterization, data analytics as a service, cloud computing**

I. Introduction

The next-generation sequencing technology has dramatically cut the DNA sequencing cost and benefited a spectrum of biosciences. For example, metagenomics, the study of the genetic material obtained directly from environmental samples to understand the microbial ecosystems [9], has thrived in recent years since various metagenome data can be generated easily. The continuous data growth, however, has imposed big burdens on the downstream data analysis.

In order to address the computational needs for metagenomic data analysis, MG-RAST [1][19][26], a metagenome annotation server, was launched in 2007 at Argonne National Laboratory as a free service and has grown into a dominant resource for the metagenomic research community worldwide. MG-RAST receives user data submissions (DNA sequence data) and processes the data with a pipeline of bioinformatics tools. As the data continue to grow, however, MG-RAST has faced pressure on productivity and the need for extra, scalable computing resources. As a result, we have built a scalable computing platform [24] and ported the entire MG-RAST pipeline into the cloud.

Using elastic cloud resources for big data analysis brings considerable flexibility. But it also introduces extra complexity and challenges regarding application-related resource utilization. For example, it is not straightforward to determine what hardware types are required for an application and how many computing cycles should be invested to serve a job. This situation is caused by the diversity of the computing resources combined with variations in the data analysis workloads. Thus, in order to efficiently utilize elastic computing resources to serve big data applications, a solid understanding of the workload characteristics is crucial. Failing to achieve such understanding may result in application failures, wastage of computing cycles, and unnecessary data movement.

In this paper, taking MG-RAST as an exemplary big data application, we characterize its workloads running in the cloud, in terms of computation, I/O, and data transfer over network. The insights of this work are expected to guide application enhancement, service operation, and resource management for MG-RAST and similar applications or services using elastic computing resources. To our best knowledge, our work is the first to characterize the cloud workloads of a production bioinformatics data analysis service. Moreover, our open-source computing and performance management platform can be applied to a variety of big data applications for scalable data analysis and workload characterization.

In the remainder of this paper, we first introduce the MG-RAST applications and the data problems (section II). Next, we briefly discuss the computing platform (section III) that runs the MG-RAST workloads in the cloud. Then, we present the workload characterization results (section IV) and discuss the implications for applications development (section V). We compare our work with some related efforts (section VI) and then summarize the paper (section VII).

II. Application Overview

In this section, we describe the MG-RAST applications as well as the data growth experienced by the MG-RAST service.

A. MG-RAST Applications

MG-RAST is a metagenomics sequence data analysis platform developed and maintained at Argonne National Laboratory [1]. MG-RAST accepts raw sequence data submission from freely registered users and automates a series of bioinformatics tools to process, analyze, and interpret the data before returning analysis results to users.

MG-RAST is popular among a number of scientific communities and has been serving users from a variety of backgrounds, such as microbiology, medical science, pharmacology, environmental science, ecology, archaeology, and anthropology. As of June 2014, MG-RAST has processed over 125,000 metagenome samples, consisting of over 50 terabase pairs (50×10^{12} bp) of sequence data for over 14,000 registered users from over 80 countries. Note that a base pair in the sequence file is represented by a character (one of A, C, G, and T, or other letters in cases of ambiguous sequences).

B. MG-RAST Pipeline: A Closer Look

An MG-RAST job, analyzing a single metagenomic data set, runs a series of data processing and analysis tasks in a pipeline (Figure 1). Basically the tasks can be categorized into three conceptual steps: quality control, data reduction, and analysis.

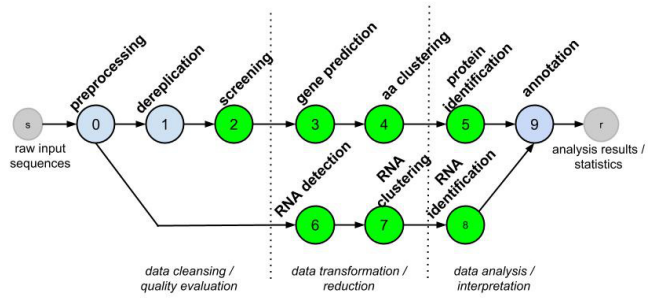


Figure 1. MG-RAST pipeline

In the quality control step, the objective is to filter out noisy sequences. "Preprocessing" removes some ambiguous sequences (with many non-ACGT characters) or low-quality sequences. "Dereplication" removes duplicated sequences, and "screening" removes unneeded human genomic sequences that may have been mixed into the sample accidentally. An "error detection" task estimates the sequence errors.

In the data reduction step, "gene prediction" finds genes in the DNA sequence. In this stage, nucleic acid sequences are transferred to amino acid sequences (from 4-letter alphabet to 20-letter alphabet). "Clustering" further compresses the data by grouping similar sequences and presenting only one "consensus" sequence for each group to the next stage.

In the data analysis step, the basic procedure is to first "identify proteins or RNAs" by querying public databases (i.e., similarity search) and then to do "annotation," assigning known functionalities to the identified protein or RNA sequences. The similarity search is computationally expensive because the databases are huge.

Some tasks use our private scripts or tools, and some use third-party tools [26]. For examples, screening uses Bowtie [17], a sequence alignment tool; gene prediction uses FragGeneScan [21], an HMM-based gene prediction tool for short and error-prone sequences; clustering uses Uclust [12], a search-based clustering tool; and the protein and RNA identification uses another sequence alignment tool named BLAT [16].

C. The Data Growth

The data submitted to MG-RAST has increased steadily over time. As shown in Figure 2, MG-RAST processed 1 Tbp of data in its first five years of operation. But in the following three years the number surged up to over 50 Tbp. After mid-2011, the data processed in each quarter is more than the total number of the first four years, and the general trend by quarter is increasing: Q3/2011 has significantly more data because in September 2011, 4.3 Tbp of data from the National Institutes of Health Human Microbiome Project [6] were submitted. Another significant increase appears recently: after Q1/2013, each month has more than 1 Tbp of data submitted—the sum of the first five years. The number of jobs submitted also keeps increasing, from thousands per year earlier to thousands per month today (Figure 3).

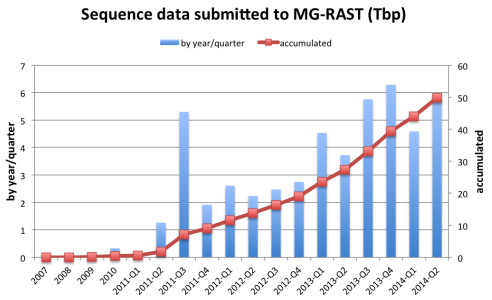


Figure 2. Sequence data (in Tbp) submitted to MG-RAST over the years/quarters. (1 Tbp = 10^{12} base pairs)

Base on MG-RAST job inventory, the input size per sample ranges from several Mbp to more than 60 Gbp. There exists two major data types in MG-RAST, 16s and shotgun. The 16s samples are typically small data sets, with average data size about 25 Mbp. The shotgun data

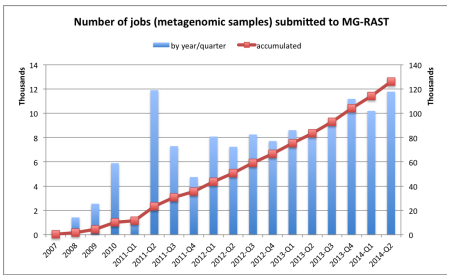


Figure 3. Number of jobs submitted to MG-RAST over the years/quarters.

is bigger, with average data size 850 Mbp and is in an increasing trend. Considering 1 base pair is represented by a character (1 $Byte$) in the text file, the file size in $Bytes$ is comparable to the number of base pairs (the former usually has larger number because the sequence file has commentary texts). Although the input file size may be moderate, the total computational cost is very high because the typical bioinformatics applications are computational expensive; that is, a small amount of data requires a long time to process. In summary, the size per sample at this scale, combined with the number of samples submitted per month, has already imposed a high cost for computation and data management.

III. Data Analysis Platform

To address the impacts by large data sets, we developed a scalable data analysis platform (Shock/AWE) that can scale out workflow computation to the cloud. The system design and implementation were discussed in our previous work [24]. Here we briefly review the system design and the deployment.

A. System Overview

Shock is a data management system for biological sequence data. It is built on on top of a backend storage system to provide object-based data store, supporting scientific metadata management, sequence file subsetting, and computational provenance. It is designed to provide convenient data query, sharing, and reuse, as well as efficient data storage for scalable and portable computing clients.

AWE is a distributed workflow and resource management system based on a server/clients model. The server receives job submissions and maintains a work queue. The AWE clients, running on heterogeneous distributed-computing resources, fetch workunits from the queue and execute them locally. AWE uses Shock for data storage and subsetting. During computation, AWE clients can report application-specific performance data to the AWE server, which is managed as job traces for online and offline study. Figure 4 shows a diagram of the Shock/AWE framework. Both Shock and AWE are open-source available at Github [7][2].

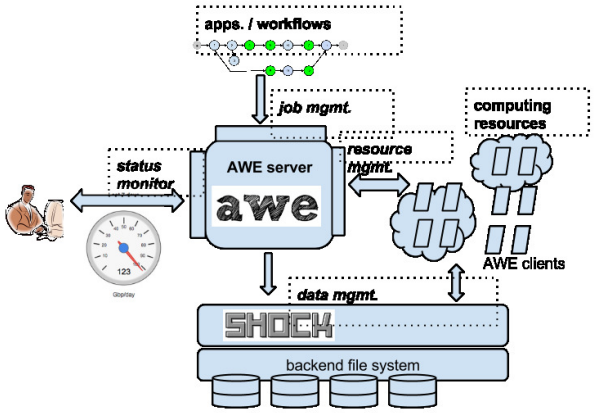


Figure 4. Shock and AWE building an integrated data analysis platform.

B. Platform Deployment

We have deployed the Shock and AWE systems in the Magellan cloud computing system[25] at Argonne. Magellan is managed by OpenStack software, tuned for scientific computing workloads. It consists of 7,500 compute cores, 30 TB of RAM, and about 1 PB of storage, connected with a QDR InfiniBand fabric. Magellan has been serving multiple research projects. For MG-RAST, we use a number of virtual machines (VMs) in Magellan as a dedicated resource pool; the number of VMs varies from 100 to 200 depending on the resource availability. We use the default configuration for all the VMs: 8 VCPUs (each with 2.6 GHz), 22 GB memory, and 300 GB disk. The platform allows us to use extra computing resources from other clouds such as Amazon EC2. We have enabled MG-RAST users to add their own resources from elsewhere to accelerate their own jobs. Here, however, we look only at the jobs processed within Magellan.

IV. Workload Characterization

We now present the workload characterization regarding compute characteristics and I/O data movement.

A. Job Trace

Our workload characterization is based on the job trace collected on the production system containing all the jobs completed in May 2014. The trace contains 4,382 complete jobs, 43,820 tasks, and 59,948 workunits. Here, a job represent a workflow run for one specific metagenome sample as input data. A task represent the computation for a particular stage in the workflow. A task can be split into one or multiple workunits, each processing a subset of the input data of the same task (Figure 5). The workunit is the unit maintained in the queue and processed by the computing clients.

As shown in Figure 1, the MG-RAST pipeline (workflow) has 10 stages, representing 10 tasks. The tasks marked in green (tasks 2 through 8) are embarrassingly parallel and

thus can be split into multiple workunits. There are two ways to split a task into workunits: by number of splits or by maximum split size. For MG-RAST tasks, we use the latter way, which means the number of splits are not fixed (depending on the total size). Table I describes the task information for the MG-RAST pipeline, including the associated bioinformatic tools and and split sizes. Note that "n/a" in the last column means the task is not split (i.e., one workunit for one task).

Table I
TASK DESCRIPTIONS.

#	Abbr.	Task Name	Tool	Split Size
0	prep	preprocessing	in-house scripts	n/a
1	depl	dereplication	in-house scripts	n/a
2	scrn	screening	Bowtie	500 MB
3	gncl	gene calling	FragGeneScan	500 MB
4	clst	protein clustering	uclust	2 GB
5	sims	protein similarity search	blat	50 MB
6	rsch	rna search	in-house scripts	n/a
7	rcls	rna clustering	uclust	2 GB
8	rsim	rna similarity search	blat	50 MB
9	annt	annotation	in-house scripts	n/a

Figure 6 shows the number of active jobs, tasks, and workunits in the job trace, from which we can get a sense of how the queue depth varies over time. For example, we see at most over 600 jobs in the system simultaneously and a peak number of over 1,200 workunits. (There are also some periods of time when the numbers are very low, which were caused by system maintenance.)

B. Computational Characteristics

We first look at the computational characteristics of MG-RAST pipeline shown in Figure 7. The compute time is per workunit processing wall time reported by AWE clients.

Figure 7(a) shows the cumulative distribute function (CDF) of the compute time consumed by each workunit, grouped by task stages. As shown in the figure, the most time-consuming task is the protein similarity search ($sims$). The median running time of each sims workunit is nearly

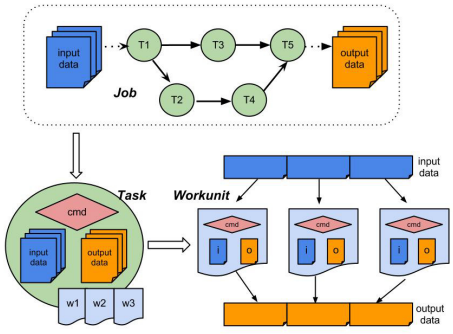


Figure 5. Key elements: job, task, and workunit

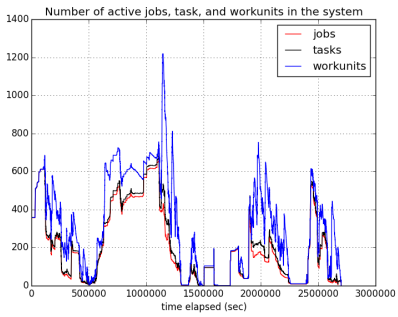


Figure 6. Number of active jobs, tasks, and workunits in the May 2014 job trace

3 hours, and the 90th percentile is about 4 hours. For other stages, the median running time per workunit is several minutes, and the 90th percentile varies from several minutes to slightly more than one hour.

Figure 7(b) shows the CDF of compute times normalized by the input data size, which represents the compute expensiveness: compute time needed per unit size of input data. As shown in the figure, for the most expensive task, 80% of workunits require 100 to 1000 seconds per megabyte; the remaining 20% need more than 1000 seconds per megabyte. For the least expensive task, $drpl$, 98% of the workunits need less than 1 second per megabyte. Other tasks fall in between: the median varies from 1 to 10 seconds per megabyte. And almost all the tasks have small and large outliers (the sharp change of the plots at the top and the bottom).

Figures 7(c) to 7(l) show the correlation between the compute time and the input data size for workunits at each stage. Each dot in the figures represents a workunit run. The x-axes are input data size in megabytes, and the y-axes represent the compute time in seconds. As shown in the figures, for some of the tasks, the workunit compute time is generally linearly related to the input size (tasks 0, 1, 2, 3, 6, 9) although each of them has some outliers.

The rest of the tasks show more complex characteristics. For example, the compute times of clustering for both protein ($clst$) and RNA ($rcls$) appear to not relate to the input size. For the most expensive task,a protein similarity search $sims$), for the same input data size, the upper bound and lower bound of the compute times increase linearly, and the gap between the bounds gets larger as input data increases. For the RNA similarity search ($rsim$), the lower bound is 0, and the upper bound increases almost linearly. For these two task, the compute times are related not only related to the input data size but also to how many matches are found between the query sequences and reference databases.

C. I/O Characteristics

In this section, we present some application characteristics related to input and output data size in order to see how data

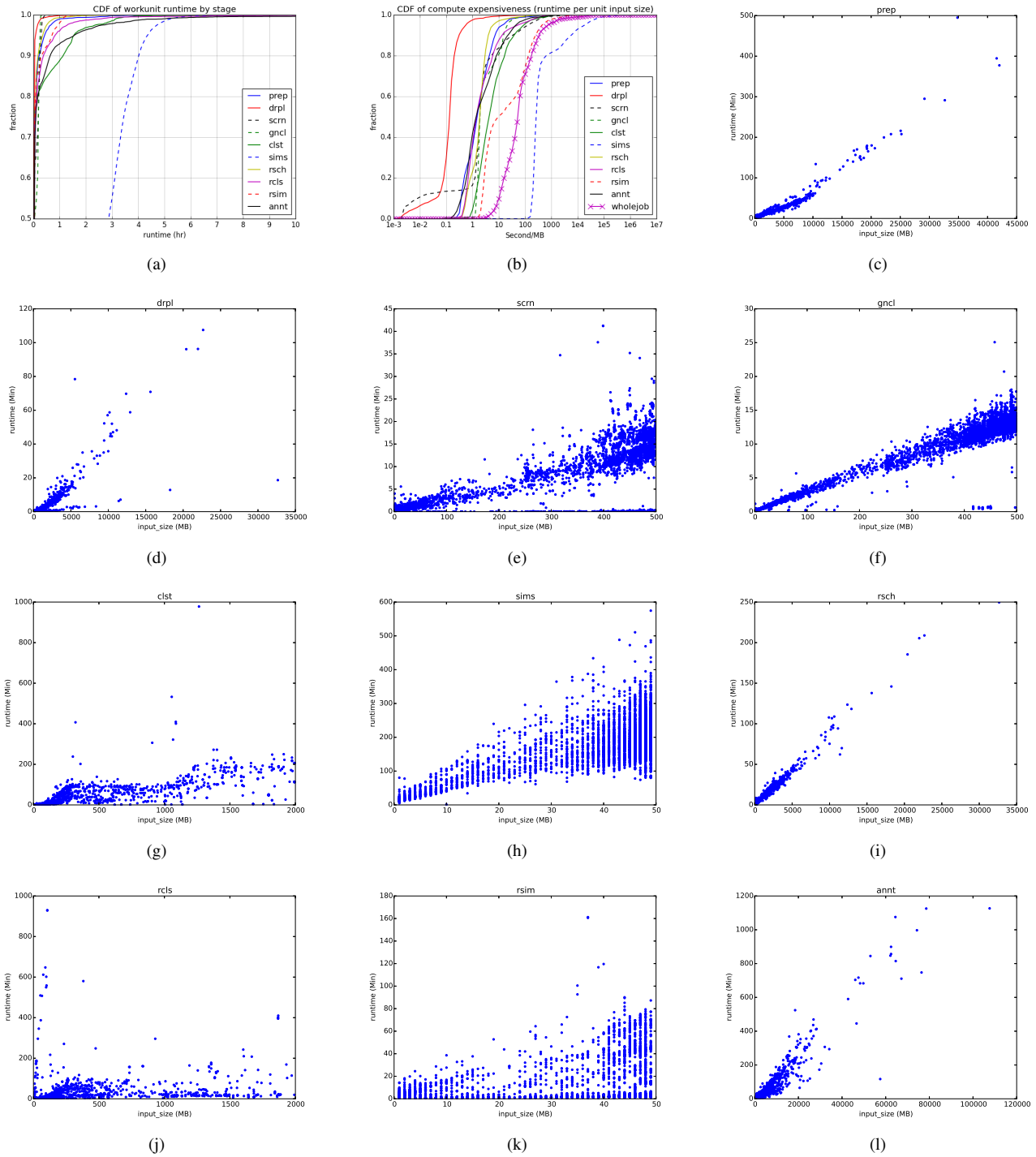


Figure 7. Compute time characteristics

gets reduced and regenerated during the pipeline computation. As shown in Figure 8, data is reduced at most of the pipeline stages. For example, the preprocessing will remove low-quality sequences and convert the raw input format from fastq [10] to fasta [3], which results in substantial data reduction. Dereplication also involves data reduction, but it retains all the sequence data (including removed ones); thus we see equal input and output data sizes. Only the passed sequences are handed into the subsequent screening stage, where we can see that the input data data is noticeably

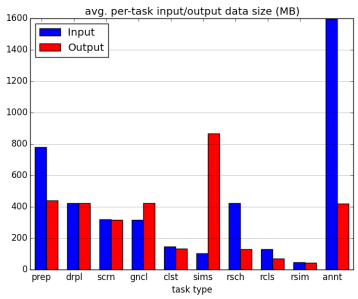


Figure 8. Characteristics of input/output data size

reduced from the last stage's output.

On the contrary, $sims$ generates more output data based on the input data sets because for each query sequence there may be multiple matching entries in the output. The $annt$ stage uses big input data sets because it uses output data sets from multiple previous stages (including raw inputs, clustering output tables, and similarity search outputs).

D. Data Movement Overhead

The side effect of scaling computation out to the cloud is the data movement cost. Therefore, we measure the data movement overhead of MG-RAST running in the cloud, in order to identify the potential problems.

Figure 9(a) shows the correlation between the data transfer time and the data size. Each dot in the figure represents an input or output data transfer between the compute clients and the data server (downloading means move data from input data server to compute clients, and uploading means move output data from compute client to the data server). The time is observed from the compute clients.

As shown in the figure, the download time increases almost linearly with the data size, except for some spikes at certain size. These spikes result when a task gets ready and is split into many workunits that are fetching the input data simultaneously, thereby causing network contentions. For example, a spike is clearly seen at 50 MB data size because the $sims$ tasks are usually split into many 50 MB-size workunits. We can also observe that data upload is much slower than data download. The reason is that in the upload process, MD5 checksum will be calculated at the data server side; thus the uploading becomes CPU-bound and suffers contention.

Both the download and upload contentions can be addressed by future enhancement of our software implementations. For example, for simultaneous downloading and uploading, a smarter data transfer scheduler is helpful; for the checksum, an asynchronized way is a potential solution. Both enhancements are out of the scope of this paper, however, which focuses on workload characterization. The main observation we get from this work is that even though there is room for further enhancement for data movement, we can achieve affordable or even negligible data movement overhead compared with the compute time, as shown in Figure 9(b) and 9(c).

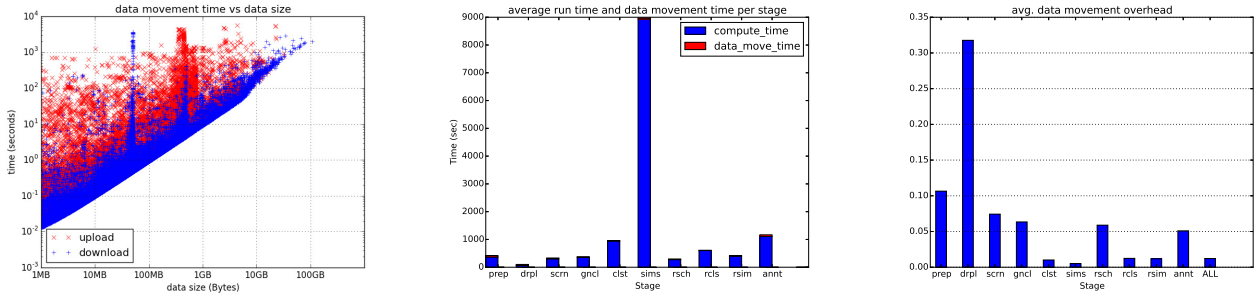
Figure 9(b) shows the average compute time and data movement time for each stage. Clearly, compared with the compute time, the data movement time is negligible. Figure 9(c) shows the average data movement overhead. The data movement overhead for a workunit is calculated by the data movement time divided by the sum of data movement time and the compute time. The average of a task stage is calculated among all the workunits belonging to the same task stage. Some stages have relatively high overhead, but the total time consumed by those stages is small, and thus the impact is trivial. The whole job's average overhead is less than 3%.

V. Discussion

From the workload characterization, we have learned that the MG-RAST, a typical instance of various bioinformatics data analysis pipelines, is composed with applications with diverse characteristics at different stages, in terms of both compute and data intensiveness. That is, for such applications, a single job requires computing resources of different features, such as hardware types, storages capabilities, and network bandwidths. Therefore, in order to utilize diverse computing resources for the various workloads, it is crucial to understand the workload characteristics at the first place, which is consistent with our motivation for this workload characterization work. Following we discuss how we can utilize the workload characterization results to benefit application development, service operation, and resource management, including some future research opportunities.

From the application development perspective, both the tool developer and the workflow developer can benefit from the performance data. The tool developer can use the performance data as feedback for each stage to identify performance bottleneck and enhance program performance, concurrency, and resource efficiency. The workflow/pipeline developer can use both the compute and I/O performance data to revise the workflow design. For example, one can choose to merge two stages involving too much data transfer in between. The performance trend can also guide task sizing and splitting. For example, in our earlier trace at the $gncl$ stage, the compute time increased sharply when the input size was more than 500 MB. Thus we chose to limit the maximum size of the $gncl$ workunit to under 500 MB.

From the service perspective, the service operator can use the performance data to estimate how long a job needs to run based on the input data size. Of course, it is hard to predict the total job turnaround time, which also depends on the queue depth, resource status, and scientific application characteristics. To provide an accurate estimation of job turnaround time is one of our future research steps. The ability to provide real-time performance data, addressed in



(a) (b) (c)

Figure 9. Data movement time and the correlation with compute time

this work, is a first step toward that direction. Also, the service operator can use the workunit runtime feedback to identify potential failures. For example, if a workunit's ratio of compute time to size is far from the expected (as shown in Figure 7, something is wrong, either with the application or with the computing resources, and time can be invested profitably to diagnose the problem.

From the resource management perspective, the workload characterization is instructive for resource allocation and capacity planning. For example, the scheduler can use the I/O and compute characteristics for data-aware task placement, shipping the compute intensive tasks to the computing resources further away from the data server and leaving the computing resources closer to the data for the I/O-intensive tasks. Also, based on the application characteristics combined with the queue status, the resource provisioner can make the right decision to scale up or down the number of computing resources of different types needed on different cloud sites. Based on this workload characterization work, data-aware scheduling and resource provisioning are on the roadmap of our future work.

VI. Related Work

MG-RAST is one of many popular biological sequence data analysis services and toolkits. For example, RAST [8] is an automated service for annotating bacterial and arterial genomes; and Galaxy [14] provides an open web-based platform for performing accessible, reproducible, and transparent genomic science. MG-RAST also provides web-based open services, but it is for metagenome analysis, which involves very different procedures from those for whole genomes [11]. Camera [23] and IMG/M [5] both implement metagenomic annotation functions, but neither has scaled to the data sizes analyzed by MG-RAST.

Computational biology research has been utilizing cloud computing systems in recent years [20][22]. Our previous work [27][28] implemented an approach to scale MG-RAST tasks into the cloud, but it was for one computationally expensive task only ($sims$) and did not support the full pipeline. Our new software is much more sophisticated, providing integrated management for applications, services, computing resources, and data products.

Understanding the workload characteristics is essential for serving the application better with the various computing systems [15]. Some public workload archives exist for the study of workload characteristics for respective computing paradigms. For example, Parallel Workload Archive [13] collects and shares job workloads from dozens of supercomputing centers worldwide; and the Grid Workloads Archive [18] provides a platform to share workloads on various grid platforms. The Google's cluster workload traces project [4] provides the trace data from Google computing cells. To our best knowledge, our work is the first to characterize the computational workloads for a production bioinformatics data analysis service running in the cloud.

VII. Summary

As the DNA sequence cost has dramatically decreased in recent years, more research opportunities have been seen in genomics and metagenomics. However, the bottleneck has been shifted to the computationally expensive data analysis; the rate of data generating by sequencing machines has been faster than the computing power improvement defined by Moore's law. Thus, utilizing distributed and elastic computing resources for bioinformatics data analysis has become a trend and, indeed, a demand. In order to efficiently use these resources, a crucial prerequisite is understanding the application workload characteristics.

In this work, we present our experience in porting MG-RAST, a production metagenomics data analysis service, into the cloud. We characterize the application characteristics from the perspective of computation, I/O, and data movement. Insights from this work can guide the further enhancement of the MG-RAST pipeline in terms of application optimization, workflow revision, service operation, and resource management. Moreover, our open-source data analysis and performance measurement framework can be applied to generic big data applications for scalable data analysis and workload characterization.

ACKNOWLEDGMENTS

This work was supported in part by the NIH award U01HG006537 "OSDF: Support infrastructure for NextGen sequence storage, analysis, and management", and U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research DE-AC02-06CH11357 as part of Resource Aware Intelligent Network Services (RAINS) and as part the Office of Science, Office of Biological and Environmental Research Systems Biology Knowledgebase (KBase).

REFERENCES

[1] MG-RAST website. http://metagenomics.anl.gov

[2] AWE project. https://github.com/MG-RAST/AWE

[3] FASTA format. http://en.wikipedia.org/wiki/FASTA_format

[4] Google's cluster workload traces https://code.google.com/p/googleclusterdata/

[5] Integrated Microbial Genomes (IMG) system. http://img.jgi.doe.gov/m

[6] NIH Human Microbiome Project. http://www.hmpdacc.org

[7] Shock project. https://github.com/MG-RAST/Shock

[8] R. K. Aziz et al., "The RAST server: Rapid annotations using subsystems technology," *BMC Genomics*, 9(1):75, 2008.

[9] N. Blow, "Metagenomics: Exploring unseen communities," *Nature*, 453:687-690, 2008.

[10] P. Cock, C. Fields, N. Goto, M. Heuer, and P. Rice, "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants,"' *Nucl. Acids Res.*, 38(6):1767-1771, 2010.

[11] N. Desai, D. Antonopoulos, J. Gilbert, E. Glass, and F. Meyer, "From genomics to metagenomics," *Current Opinion in Biotechnology*, 23:72-76, 2012.

[12] R. C. Edgar. "Search and clustering orders of magnitude faster than BLAST," *Bioinformatics*, 26(19):2460-1, 2010.

[13] D. Feitelson, D. Tsafrir, D. Krakov, "Experience with the parallel workloads archive," 2012.

[14] J. Goecks, A. Nekrutenko, J. Taylor, et al., "Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life science," *Genome Biology*, 11:R86, 2010.

[15] L. John, P. Vasudevan, and J. Sabarinathan, "Workload characterization: motivation, goals and methodology," in *Workload Characterization: Methodology and Case Studies*, 3-14, 1999.

[16] W. J. Kent, "BLAT–the BLAST-like alignment tool," *Genome Research*, 12(4):656-664, 2002.

[17] B. Langmead, C. Trapnell, M. Pop, and S. L. Sailzberg. "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*. 10:R25, 2009.

[18] A. Losup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. Epema, "The Grid Workloads Archive," University of Delft, 2008.

[19] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. M. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening, and R. Edwards, "The metagenomics RAST server–a public resource for the automatic phylogenetic and functional analysis of metagenomes," *BMC Bioinformatics*, 9(386):1-8, 2008.

[20] N. Mohamed, H. Lin, and W.-C. Feng, "Accelerating data-intensive genome analysis in the cloud," in *Proc. of International Conference on Bioinformatics and Computational Biology (BICoB)*, 2013.

[21] M. Rho, H. Tang, and Y. Ye, "FragGeneScan: Predicting genes in short and error-prone reads," *Nucl. Acids Res.*, 2010. doi:10.1093/nar/gkq747

[22] J. Qiu, J. Ekanayake, T. Gunarathne, J. Y. Choi, S.-H. Bae, H. Li, B. Zhang, T.-L. Wu, Y. Ruan, S. Ekanayake, A. Hughes, and G. Fox, "Hybrid cloud and cluster computing paradigms for life science applications," *BMC Bioinformatics*, 11(12):53, 2010.

[23] S. Sun, J. Chen, W. Li, I. Altinatas, A. Lin, S. Peltier, K. Stocks, E. Allen, M. Ellisman, J. Grethe, and J. Wooley, "Community cyberinfrastructure for advanced microbial ecology research and analysis: The CAMERA resource," *Nucl. Acids Res.*, 39:D546-551, 2010.

[24] W. Tang, J. Wilkening, N. Desai, W. Gerlach, A. Wilke, and F. Meyer, "A scalable data analysis platform for metagenomics," in *Proc. of IEEE International Conference on Big Data*, 2013.

[25] L.Ramakrishnan, P. Zbiegel, S. Campbell, R. Bradshaw, R. S. Canon, S. Coghlan, I. Sakrejda, N. Desai, T. Declerck, and A. Liu, "Magellan: Experiences from a science cloud," in *Proc. of ACM International Workshop on Scientific Cloud Computing*, 2011.

[26] A. Wilke, E. Glass, D. Bartels, J. Bischof, D Braithwaite, M. D'Souza, W. Gerlach, T. Harrison, K. Keegan, H. Matthews, R. Kottmann, T. Paczian, W. Tang, W. Trimble, P. Yilmaz, J. Wilkening, N. Desai, and F. Meyer, "A metagenomics portal for a democratized sequencing world," *Methods in Enzymology*, 531:487-523, 2013.

[27] A. Wilke, J. Wilkening, E. Glass, N. Desai, and F. Meyer, "An experience report: Porting the MG-RAST rapid metagenomics analysis pipeline to the cloud," *Concurrency and Computation: Practice and Experience*, 23:2250-2257, 2011.

[28] J. Wilkening, A. Wilke, N. Desai, and F. Meyer, "Using clouds for metagenomics: A case study," in *Proc. of IEEE International Conference on Cluster Computing*, 2009.